CSC 482 HW#1 Problem 1.1

Kefu Zhu, kzhu6

Objective:

We are given n positive numbers $a_1, \ldots, a_n$. The goal is to select a subset of the numbers with maximal sum and such that no three consecutive numbers are selected with a $O(n)$-time algorithm

Define:

- $numList$ = the input list that contains n positive numbers
- $maxSum[i]$ = the maximum sum of a subset of the first i items (items $1, 2, \ldots i$), such that no three elements are consecutive

Sudo code:

```
# Base case
maxSum[1] = numList[1]
maxSum[2] = numList[1] + numList[2]
maxSum[3] = max(numList[1] + numList[2], # Do not use numList[3]
                numList[1] + numList[3], # Do not use numList[2]
                numList[2] + numList[3]) # Do not use numList[1]

# Dynamic programming for the rest of elements
# Given the optimal ansewr: maxSum[1], maxSum[2], ..., maxSum[i-1]
# Three general cases for maxSum[i]:
# (1) Do not use element i
# (2) Do not use element i-1
# (3) Do not use element i-2

# Pick the maximum value from these three cases and store it in maxSum[i]
maxSum[i] = max(maxSum[i-1],
                maxSum[i-2] + numList[i],
                maxSum[i-3] + numList[i-1] + numList[i])
```

Python code:

```python
def maxSum(numlist):
    # Initialize an empty list to store the maximum sum of subarray numlist[0..i], such
    _sum = [None]*len(numlist)

    # Initialization of basic scenarios
    if len(numlist) >= 1:
        _sum[0] = numlist[0]
    if len(numlist) >= 2:
        _sum[1] = numlist[0] + numlist[1]
    # Base case: we have three consecutive elements at the first time
    if len(numlist) > 2:
        _sum[2] = max(numlist[0]+numlist[1],
                      numlist[0]+numlist[2],
                      numlist[1]+numlist[2])
    # Dynamic programming for the rest of elements
    for i in range(3, len(numlist)):
        _sum[i] = max(_sum[i-1],
                      _sum[i-2] + numlist[i],
                      _sum[i-3] + numlist[i-1] + numlist[i]
                     )
        # Increment the index
        i += 1

    # Return the max sum given all elements in the input list
    return _sum[-1]
```

```python
# Test cases
num_list1 = [5,5,8,5,5]
num_list2 = [5,5,12,5,5]
num_list3 = [1,2,2,1,2,1,2,5,5]
```

```python
maxSum(num_list1) # Output: 20
maxSum(num_list2) # Output: 22
maxSum(num_list3) # Output: 17
```